

# PROYECTO FINAL REDES E INFRAESTRUCTURA

Autores: Ana Cristina Quintero Carpintero, María Fernanda Tello Vergara, Sebastián Belalcázar Mosquera.

Profesor: Oscar Hernán Mondragón Martínez.

*Ingeniería De Datos e Inteligencia Artificial, Universidad Autónoma De Occidente.*

Mayo 24, 2024.

**RESUMEN:** El presente proyecto final tiene como objetivo el diseño e implementación de una infraestructura basada en microservicios y contenedores para el análisis de datos académicos de estudiantes de East High. Se utilizarán tecnologías como Docker, Docker Swarm, Apache Spark y PySpark para lograr un sistema escalable y eficiente, que permita procesar grandes volúmenes de datos y generar reportes visualizables en un dashboard.

## INTRODUCCIÓN

El análisis de datos en entornos educativos permite obtener información valiosa para mejorar las estrategias pedagógicas y las

políticas desarrollará una infraestructura que permita realizar el proceso de contenedores y desplegar aplicaciones de análisis de datos en un clúster de procesamiento distribuido. Se utilizarán herramientas como Docker para los contenedores, Apache Spark para el procesamiento de datos y Flask para la creación de APIs y dashboards.

## REPOSITORIO

El repositorio del proyecto está disponible en GitHub:

<https://github.com/SEBASBELMOS/Caoskol-Project>

## DOCKER COMPOSE

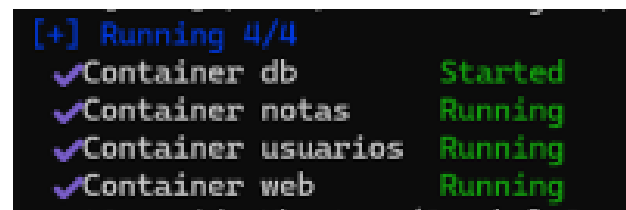
Git clone  
<https://github.com/SEBASBELMOS/Caoskol-Project>

cd Caoskol-Project

docker compose up --build -d

Verificar los contenedores en ejecución:

Docker ps



```
[+] Running 4/4
✓ Container db      Started
✓ Container notas   Running
✓ Container usuarios Running
✓ Container web     Running
```

## PRUEBA FUNCIONAMIENTO

Docker Compose: Acceder a <http://192.168.100.3:8090/> en un navegador.

Docker Swarm: Acceder a <http://192.168.100.3:9080/> en un navegador.

**East High**  
Resultados de Pruebas Anuales

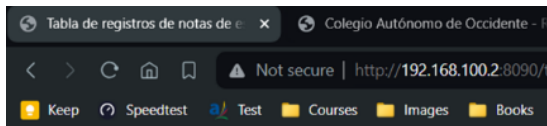
---

**Iniciar Sesión**

ID

Clave

**Ingresar**



## Estadísticas notas de estudiantes

Estudiante	Grado	Nota de Matemáticas	Nota de Inglés
Ana Quintero	10	85	78
MarAa Fernanda Tello	11	90	82
Sebastian Belcazar	10	90	90
Lewis Hamilton	9	88	85
Carlos Sainz	8	80	87

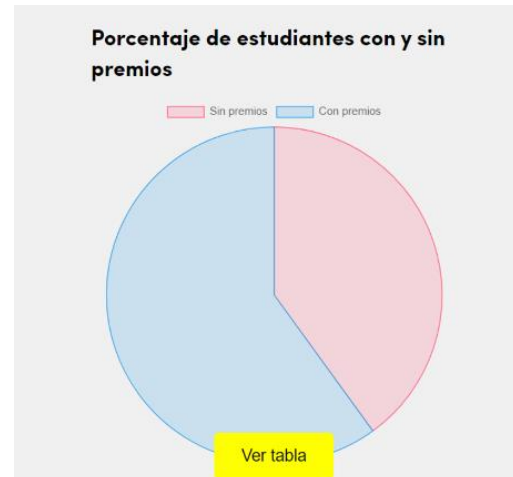
**Bienvenido**

**Cerrar Sesión**

**Sebastian Belcazar**

**Grado: 10**

<b>Número de Premios</b> 0	<b>Puntaje de Matemáticas</b> 90
<b>Puntaje de Inglés</b> 90	<b>Puntaje de Ciencias</b> 80



## DOCKER SWARM

**Cliente: VM Principal.**

```
docker swarm leave --force
```

```
docker swarm init --advertise-addr  
192.168.100.3
```

```
docker stack deploy -c docker-swarm.yml  
caoskol
```

```
Creating network caoskol_app-network  
Creating service caoskol_notas  
Creating service caoskol_web  
Creating service caoskol_haproxy  
Creating service caoskol_db  
Creating service caoskol_usuarios
```

**Servidor: VM Secundaria.**

```
docker swarm leave --force
```

```
docker swarm join --token [TOKEN]  
192.168.100.2:2377
```

## CREACIÓN Y GESTIÓN DE IMÁGENES

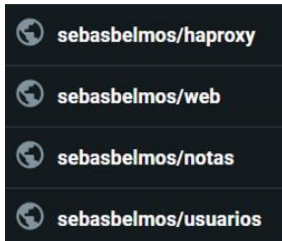
```
docker build -t sebasbelmos/usuarios:latest  
./appColegio/microUsuarios
```

```
docker push sebasbelmos/usuarios:latest
```

```
docker build -t sebasbelmos/notas:latest
./appColegio/microNotas
```

```
docker push sebasbelmos/notas:latest
```

```
docker build -t sebasbelmos/web:latest
./APPWEB
```



## Estado del Swarm y servicios

Verificar los nodos:

```
docker node ls
```

```
root@servidorUbuntu:~/Caoskol-Project# docker node ls
ID                HOSTNAME          STATUS
bovw8gpns9mab5uv3esft650v  clienteUbuntu    Ready
2du9jcnfjh68c8wpmipkgq90h * servidorUbuntu    Ready
root@servidorUbuntu:~/Caoskol-Project#
```

## Verificar los servicios desplegados:

```
docker service ls
```

```
root@clienteUbuntu:~/Caoskol-Project# docker service ls
ID                NAME                MODE                REPLIC
4vwhwndhnude     caoskol_db          replicated          1/1
s7z0rbm8oq9a     caoskol_haproxy     replicated          1/1
p87xdp2za1jb     caoskol_notas       replicated          2/2
omg1ck66urc      caoskol_usuarios    replicated          2/2
54oxp4imi281     caoskol_web         replicated          3/3
root@clienteUbuntu:~/Caoskol-Project#
```

```
root@clienteUbuntu:~/Caoskol-Project#
caoskol_web scaled to 10
overall progress: 10 out of 10 tasks
1/10: running [=====]
2/10: running [=====]
3/10: running [=====]
4/10: running [=====]
5/10: running [=====]
6/10: running [=====]
7/10: running [=====]
8/10: running [=====]
9/10: running [=====]
10/10: running [=====]
verify: Service caoskol_web converged
root@clienteUbuntu:~/Caoskol-Project#
```

```
docker push sebasbelmos/web:latest
```

```
docker build -t sebasbelmos/haproxy:latest
./haproxy
```

```
docker push sebasbelmos/haproxy:latest
```

## Apagar el cluster:

```
docker stack rm caoskol
```

## Configuración de HAproxy

```
docker config create haproxy_config
haproxy.cfg
```

```
docker config rm haproxy_config
```

## Logs de servicios:

```
docker service logs container_name
```

## ESCALABILIDAD DE SERVICIOS

```
docker service scale caoskol_web=10
```

```
docker service scale caoskol_web=1
```

```
root@clienteUbuntu:~/Caoskol-Project#
caoskol_web scaled to 1
overall progress: 1 out of 1 tasks
1/1:
verify: Service caoskol_web converged
root@clienteUbuntu:~/Caoskol-Project#
```

PRUEBAS CON JMETER

HTTP Request

HTTP Request

Name: HTTP Request

Comments:

Basic

Advanced

Web Server

Protocol [http]: Server Name or IP: 192.168.100.3

HTTP Request

GET Path:

☐ Redirect Automatically

☒ Follow Redirects

☒ Use KeepAlive

☐ Use multipart

View Results Tree

Text

HTTP Request

HTTP Request

HTTP Request

HTTP Request

HTTP Request

HTTP Request

HTTP Request

HTTP Request

HTTP Request

HTTP Request

Sampler result

Request

Response data

Thread Name:grupoPrueba 1-1

Sample Start:2024-05-23 18:46:45 COT

Load time:1

Connect Time:0

Latency:1

Size in bytes:3707

Sent bytes:123

Headers size in bytes:308

Body size in bytes:3399

Sample Count:1

Error Count:0

Data type ("text"|"bin"|""):text

Response code:200

Response message:OK

HTTPSampleResult fields:

ContentType: text/html

DataEncoding: null

Summary Report

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename:

Label	# Samples	Average	Min	Max	Std. Dev.
HTTP Request	10	6	1	48	13.81
TOTAL	10	6	1	48	13.81

Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename:

Label	# Samples	Average	Median	90% Line	95% Line
HTTP Request	10	6	2	3	3
TOTAL	10	6	2	3	3

Aggregate Report

Browse... Log/Display Only: <input type="checkbox"/> Errors <input type="checkbox"/> Successes <input type="checkbox"/> Configure						
99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
48	1	48	0.00%	137.0/sec	495.92	16.45
48	1	48	0.00%	137.0/sec	495.92	16.45

El porcentaje de error disminuye entre más instancias se creen.

A más peticiones más errores.

PROCESAMIENTO Y ANÁLISIS CON PYSPARK

Configuración del master

```
./start-master.sh --host 192.168.100.3 --port 7077 --webui-port 8080
```

Spark 3.5.1 Spark Master at spark://192.168.100.3:7077

URL: spark://192.168.100.3:7077

Active Workers: 0

Cores in use: 0 Total: 0 Used:

Memory in use: 0.0 GB Total: 0.0 GB Used:

Resources in use:

Applications: 0 Running 3 Completed

Drivers: 0 Running 0 Completed

Status: ALIVE

Workers (0)

Worker Id	Address	State	Cores	Memory	Resources
-----------	---------	-------	-------	--------	-----------

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240524030628-0000	ConsoleAnalysis	0	2.0 GB		2024/05/24 03:06:28	root	FINISHED	24 s
app-20240524030629-0001	ConsoleAnalysis	0	10240 MB		2024/05/24 03:06:29	root	FINISHED	18 s
app-20240524030632-0000	ConsoleAnalysis	0	10240 MB		2024/05/24 03:06:32	root	FINISHED	2 s

Ejemplo de análisis en PySpark



```
df = spark.read.options(header='True', inferSchema='True').csv('/root/Caoskol-Project/US_Dataset.csv')
```

```
df.count()
```

```
>>> df.count()
400
```

```
df.printSchema()
```

```
>>> df.printSchema()
root
 |-- id: integer (nullable = true)
 |-- nombre: string (nullable = true)
 |-- grado: integer (nullable = true)
 |-- num_premios: integer (nullable = true)
 |-- programa: string (nullable = true)
 |-- puntaje_mat: integer (nullable = true)
 |-- puntaje_ing: integer (nullable = true)
 |-- puntaje_ciencias: integer (nullable = true)
```

```
df.select("grado").distinct().show()
```

```
>>> df.select("grado").distinct().show()
+-----+
| grado |
+-----+
|      12|
|       6|
|       9|
|       8|
|       7|
|      10|
|      11|
+-----+
```

EJECUCIÓN DE ARCHIVO [appPySpark.py](#)

```
./spark-submit /root/Caoskol-Project/appPySpark.py
```

```
root@clienteUbuntu:~/Caoskol-Project# ls
appColegio db LICENSE 'Pro
appPySpark.py docker-compose.yml node_modules res
APPWEB docker-swarm.yml package.json US
dataset.py haproxy package-lock.json US
root@clienteUbuntu:~/Caoskol-Project# cd results.csv/
root@clienteUbuntu:~/Caoskol-Project/results.csv# ls
part-00000-d1b3151b-bdbb-4f7e-9b9e-d4402c1e9776-c000.csv _S
```

Prueba:

```
root@clienteUbuntu:~/Caoskol-Project# cd results.csv/
root@clienteUbuntu:~/Caoskol-Project/results.csv# ls
part-00000-d1b3151b-bdbb-4f7e-9b9e-d4402c1e9776-c000.csv
root@clienteUbuntu:~/Caoskol-Project/results.csv# spark-submit ./appPySpark.py
grado, count
7, 76
11, 47
8, 52
6, 86
9, 48
10, 61
12, 30
NULL, 3
```

## ANÁLISIS DE RESULTADOS

Este script en PySpark realiza un análisis de datos cargando un archivo CSV, limpiando los nombres de las columnas para eliminar

espacios y caracteres especiales, y calculando diversas estadísticas descriptivas como medias, desviaciones estándar, máximos y mínimos para puntajes en matemáticas, inglés, ciencias y número de premios. También agrupa los datos para calcular la distribución de estudiantes y promedios de puntajes por grado, y determina la correlación entre el número de premios y los puntajes. Finalmente, guarda los resultados en archivos CSV y cierra la sesión de Spark.

EJECUCIÓN DE ARCHIVO [analysis.py](#)

```
./spark-submit /root/Caoskol-Project/analysis.py
```

```
root@clienteUbuntu:~/Caoskol-Project# ls
analysis.py clean_dataset_users.py
appColegio correlacion_results
appPySpark.py db
APPWEB docker-compose.yml
clean_dataset.py docker-swarm.yml
root@clienteUbuntu:~/Caoskol-Project#
```

Cómo podemos ver en la carpeta del proyecto quedan nuevas carpetas en donde se pueden ver los resultados:

```

+-----+-----+-----+-----+
|Promedio Matemáticas|Promedio Inglés|Promedio Ciencias|Promedio Premios|
+-----+-----+-----+-----+
|Estándar Premios|Máximo Premios|Mínimo Premios|
+-----+-----+-----+-----+
| 60.3775| 60.735| 0.0| 1.5725|
+-----+-----+-----+-----+
| 0.829013983722113| 3.0| 0.0|
+-----+-----+-----+-----+
24/05/24 12:45:09 INFO FileSourceStrategy: Pushed Filters:
```

```

+-----+-----+-----+-----+
|Desvío Estándar Matemáticas|Desvío Estándar Inglés|Desvío Estándar Ciencias|Desvío
+-----+-----+-----+-----+
| 23.92752200968185| 22.087766945111348| NULL| 1.
+-----+-----+-----+-----+
```

```

+-----+-----+-----+-----+
|max_math|min_math|avg_math|max_awards|avg_awards|
+-----+-----+-----+-----+
| 100.0| 20.0| 60.3775| 3.0| 1.5725|
+-----+-----+-----+-----+
```

grado	count
8.0	52
7.0	76
11.0	47
10.0	61
6.0	86
9.0	48
12.0	30

grado	Promedio Matemáticas	Promedio Inglés	Promedio Ciencias
8.0	63.09615384615385	59.44238769238769	NULL
7.0	57.85526315789474	60.671852631578945	NULL
11.0	57.97872340425532	65.57446888510639	NULL
10.0	58.78688524590164	61.114754098360656	NULL
6.0	62.93023255813954	60.05813953488372	NULL
9.0	62.166666666666664	58.291666666666664	NULL
12.0	58.866666666666667	60.63333333333333	NULL

Correlación Premios-Matemáticas	Correlación Premios-Inglés	Correlación Premios-Ciencias
-0.02141959461403...	-0.04184899492989237	NULL

## ANÁLISIS DE RESULTADOS

Este script en PySpark comienza cargando un archivo CSV con datos de estudiantes y limpiando los nombres de las columnas para eliminar caracteres problemáticos. Posteriormente, se calculan diversas estadísticas descriptivas, incluyendo promedios, desviaciones estándar, valores máximos y mínimos para los puntajes en matemáticas, inglés, ciencias y el número de premios. El análisis también incluye la distribución de estudiantes por grado y los promedios de puntajes por cada grado. Además, se calcula la correlación entre el número de premios y los puntajes en diferentes materias. Los resultados de estas operaciones se guardan en archivos CSV, y la sesión de Spark se cierra al final del proceso.

## ANÁLISIS DEL DATASET

La selección de un dataset adecuado es crucial para garantizar la calidad y relevancia de los resultados obtenidos a partir del análisis de datos. En este proyecto, hemos optado por utilizar la

base de datos de los estudiantes de East High, que incluye información detallada sobre las notas y premios obtenidos por aproximadamente 400 estudiantes en distintas asignaturas como matemáticas, inglés y ciencias. Esta base de datos también contiene información sobre programas de estudio específicos en los que los estudiantes pueden estar inscritos, aunque estos datos no son visibles por políticas de seguridad.

La elección de este dataset se justifica por las siguientes razones:

1. **Relevancia y Cobertura:** Proporciona una visión integral del rendimiento académico de los estudiantes en diversas áreas clave, permitiendo análisis comparativos y longitudinales.
2. **Riqueza de Datos:** Incluye no solo las notas de los estudiantes, sino también información sobre premios y programas de estudio, lo que permite realizar análisis más complejos sobre los factores que influyen en el rendimiento académico.
3. **Confidencialidad:** Asegura la privacidad de los datos sensibles, alineándose con las políticas de seguridad y protección de datos.

## Beneficios del Análisis de Datos

El análisis del dataset de East High permite:

- **Identificación de Factores de Éxito:** Determinar qué variables están asociadas con el alto rendimiento académico y la obtención de premios.
- **Evaluación de Programas Educativos:** Analizar el impacto de los diferentes programas de estudio en los resultados de los estudiantes.
- **Predicción de Rendimiento:** Utilizar modelos predictivos para anticipar el rendimiento futuro de los estudiantes y sus necesidades de apoyo.
- **Mejora de Políticas Educativas:** Informar las decisiones sobre estrategias pedagógicas y políticas

educativas basadas en datos empíricos.

### Alternativas de Solución y Justificación

Para el empaquetado y despliegue de la aplicación, se evaluaron diferentes tecnologías y herramientas disponibles. Finalmente, se decidió utilizar la plataforma de procesamiento de datos distribuidos Apache Spark debido a sus ventajas en términos de rendimiento, escalabilidad, flexibilidad y capacidades avanzadas de procesamiento en memoria y en tiempo real. PySpark, la API de Python para Apache Spark, se utilizará para manejar eficientemente grandes volúmenes de datos y realizar análisis complejos de manera efectiva.

La aplicación generará reportes que serán visualizados en un dashboard interactivo. Para la creación del dashboard, se consideraron varias herramientas de visualización, optándose por aplicaciones usando el lenguaje JavaScript y php debido a su simplicidad y flexibilidad para desarrollar aplicaciones web.

### DISEÑO

1. Propuesta del pipeline, componentes o algoritmos a utilizar: se deben definir los componentes y servicios que se deben implementar, y los algoritmos y pipelines que se deben utilizar para el procesamiento de datos.

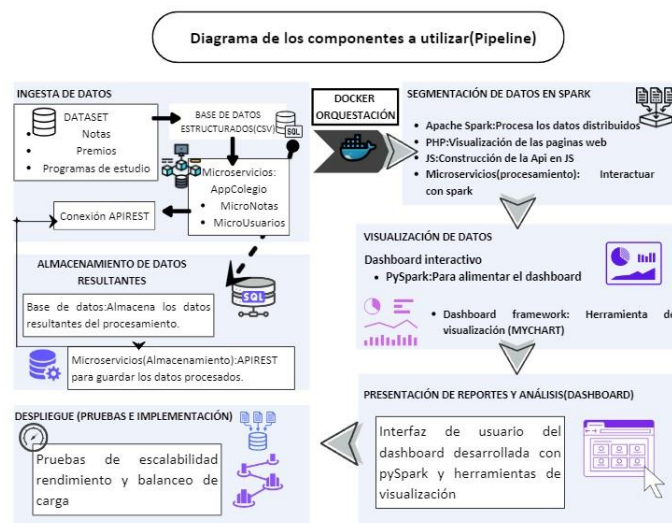
- Diagrama de los componentes a utilizar: se debe crear un diagrama que muestre los componentes que se van a utilizar en el proyecto y cómo se relacionan entre sí.

- Relación y flujo de trabajo entre los componentes: se debe definir cómo los diferentes componentes interactúan entre sí y cómo fluye el trabajo a través del sistema.

- Descripción de los componentes: se debe proporcionar una descripción detallada de cada uno de los componentes del sistema, incluyendo su función y los datos que manejan.

2. Diagrama de despliegue: se debe crear un diagrama que muestre cómo se despliegan los diferentes componentes en el sistema, incluyendo los nodos en los que se ejecutan.

### 1. Componentes y Servicios a Implementar (PIPELINE)





Para el proyecto de análisis de datos de los estudiantes del East High, se utilizarán los siguientes componentes y servicios:

### **Contenedores y Orquestación:**

*Docker:* Para contener los microservicios y asegurar que cada componente sea independiente y fácilmente desplegable.

### **Base de Datos:**

*Base de datos SQL:* Para almacenar datos estructurados sobre estudiantes, notas y premios.

### **Procesamiento de Datos Distribuido:**

*Apache Spark:* Para el procesamiento de datos en un entorno distribuido.

*PySpark:* API de Python para interactuar con Apache Spark y realizar análisis de datos.

### **Visualización de Datos:**

*JavaScript:* Para servir el dashboard de visualización.

*Jupyter Notebooks:* Para desarrollo y pruebas interactivas.

### **CI/CD Pipeline:**

*GitHub:* Para la integración y el despliegue continuo, asegurando que los cambios en el código se prueben y desplieguen automáticamente.

## **RELACIÓN Y FLUJO DE TRABAJO ENTRE LOS COMPONENTES**

**Datos de Entrada:** Los datos de los estudiantes (notas, premios, programas de estudio) son almacenados en la base de datos.

**Extracción y Procesamiento de Datos:** Un microservicio desarrollado con en JavaScript extrae los datos relevantes de la base de datos. Los datos extraídos son enviados a Apache Spark para procesamiento mediante PySpark.

**Análisis y Generación de Reportes:** Apache Spark procesa los datos, realiza análisis complejos y genera resultados. Los resultados del análisis son enviados a otro microservicio que formatea estos datos para visualización.

**Visualización de Datos:** Los datos procesados se visualizan en un dashboard interactivo creado desde el código de programación. Los usuarios pueden interactuar con el dashboard para ver reportes detallados.

**Despliegue y Orquestación:** Todos los microservicios y componentes son contenerizados usando Docker.

**CI/CD Pipeline:** GitHub automatizan la integración y el despliegue, probando cada cambio en el código y desplegando los servicios.

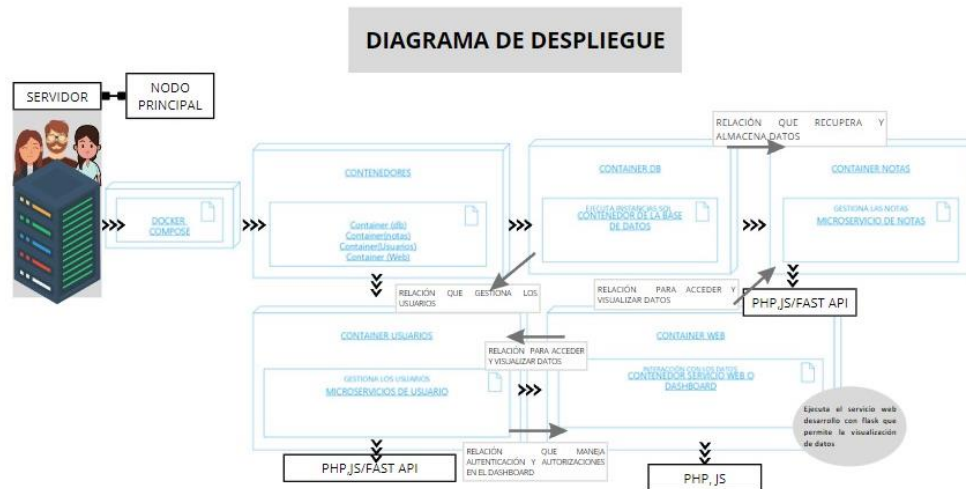
Esta estructura asegura un flujo de trabajo eficiente y escalable para el análisis de datos, desde la extracción hasta la visualización, garantizando que cada componente funcione de manera independiente

## **DIAGRAMA DE DESPLIEGUE**

El diagrama de despliegue muestra cómo se distribuyen los componentes del sistema en los nodos de hardware y los contenedores que se ejecutan en cada uno.

El diagrama de despliegue puede ser representado de la siguiente manera:





## EXPLICACIÓN DIAGRAMA:

**Servidor Ubuntu:** Este es el nodo principal donde se ejecuta el Docker Engine. **Docker Engine:** Motor de contenedores que gestiona la ejecución de contenedores. **Contenedores:**

- **DB:** Ejecuta una instancia de MySQL para la base de datos.
- **NOTAS:** Ejecuta un microservicio desarrollado en JavaScript para la gestión de notas.
- **USUARIOS:** Ejecuta un microservicio desarrollado en JavaScript para la gestión de usuarios.
- **WEB:** Ejecuta el servicio web desarrollado en JavaScript y php que sirve el dashboard para la visualización de datos.

## ALTERNATIVAS DE SOLUCIÓN

*Implementación de Tecnologías de Orquestación Alternativas:* Además de Docker Swarm, podrías considerar tecnologías de orquestación como Kubernetes. Kubernetes ofrece características avanzadas de gestión de contenedores y escalabilidad que podrían ser beneficiosas para tu infraestructura.

*Integración de Machine Learning:* Si el objetivo es mejorar aún más la capacidad predictiva del sistema, podrías integrar

técnicas de aprendizaje automático utilizando bibliotecas como scikit-learn o TensorFlow. Esto te permitiría desarrollar modelos predictivos para identificar patrones en los datos estudiantiles y realizar predicciones más precisas sobre el rendimiento académico.

*Mejora de la Interfaz de Usuario:* Recopilar retroalimentación de los usuarios finales y realizar mejoras continuas en la interfaz de usuario del dashboard para hacerla más intuitiva, fácil de usar y orientada a las necesidades específicas de los usuarios.

## CONCLUSIONES

Este proyecto demuestra la viabilidad de utilizar una infraestructura basada en microservicios y contenedores para el análisis de datos educativos. La utilización de Docker, Apache Spark y PySpark permite un procesamiento eficiente y escalable de grandes volúmenes de datos, facilitando la generación de reportes y su visualización en un dashboard interactivo.

## APRENDIZAJES Y RESULTADOS CLAVE

- **Escalabilidad y Eficiencia:** Docker y Docker Swarm proporcionan una solución escalable y eficiente para desplegar microservicios.

- **Procesamiento Distribuido:** Apache Spark y PySpark son herramientas poderosas para el procesamiento distribuido de grandes volúmenes de datos.
- **Flexibilidad y Agilidad:** La utilización de JavaScript y php para

el desarrollo de APIs y dashboards permite una gran flexibilidad en la creación de interfaces de usuario.

- **Automatización:** GitHub CI/CD facilita la integración y despliegue continuo, mejorando la eficiencia y reduciendo el riesgo de errores.